🛠️💻Crowd Funding App

Data Storage

```
import os
from typing import Dict, List

users_data = {}
projects_data = []
current_user_email = None

USERS_FILE = 'users.txt'
PROJECTS_FILE = 'projects.txt'

def load_data():
    global users_data, projects_data

    if os.path.exists(USERS_FILE):
        try:
            with open(USERS_FILE, 'r', encoding='utf-8') as f:
                users_data = {}
                for line in f:
                    line = line.strip()
                    if line:
                        parts = line.split('|')
                        if len(parts) == 7:
                            email = parts[0]
                            users_data[email] = {
                                'first_name': parts[1],
                                'last_name': parts[2],
                                'email': email,
                                'password_hash': parts[3],
                                'mobile': parts[4],
                                'is_active': parts[5].lower() == 'true',
```

```
                        'created_at': parts[6]
                    }
        except:
            users_data = {}

    if os.path.exists(PROJECTS_FILE):
        try:
            with open(PROJECTS_FILE, 'r', encoding='utf-8') as f:
                projects_data = []
                for line in f:
                    line = line.strip()
                    if line:
                        parts = line.split('|')
                        if len(parts) == 8:
                            projects_data.append({
                                'title': parts[0],
                                'details': parts[1],
                                'total_target': float(parts[2]),
                                'start_date': parts[3],
                                'end_date': parts[4],
                                'owner_email': parts[5],
                                'created_at': parts[6],
                                'current_amount': float(parts[7])
                            })
        except:
            projects_data = []

def save_data():
    with open(USERS_FILE, 'w', encoding='utf-8') as f:
        for email, user in users_data.items():
            line = f"{user['email']}|{user['first_name']}|{user['last_name']}|{user['p
assword_hash']}|{user['mobile']}|{user['is_active']}|{user['created_at']}\n"
            f.write(line)

    with open(PROJECTS_FILE, 'w', encoding='utf-8') as f:
        for project in projects_data:
```

```
        line = f"{project['title']}|{project['details']}|{project['total_target']}|{project['start_date']}|{project['end_date']}|{project['owner_email']}|{project['created_at']}|{project['current_amount']}\n"
        f.write(line)

def get_current_user():
    global current_user_email
    if current_user_email and current_user_email in users_data:
        return users_data[current_user_email]
    return None

def set_current_user(email):
    global current_user_email
    current_user_email = email

def clear_current_user():
    global current_user_email
    current_user_email = None
```

Validation

```
import re
from datetime import datetime

def validate_egyptian_phone(phone):
    patterns = [
        r'^(\+20|0020|20)?01[0125][0-9]{8}$',  # Mobile numbers
        r'^(\+20|0020|20)?02[0-9]{8}$'         # Landline numbers
    ]
    phone = re.sub(r'[\s\-\(\)]', '', phone)
    return any(re.match(pattern, phone) for pattern in patterns)

def validate_email(email):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(pattern, email) is not None
```

```
def validate_date(date_str):
    try:
        datetime.strptime(date_str, '%Y-%m-%d')
        return True
    except ValueError:
        return False

def validate_name(name):
    if not name or not name.strip():
        return False

    return all(char.isalpha() or char.isspace() for char in name.strip()) and name.
strip()

def validate_date_range(start_date, end_date):
    try:
        start_dt = datetime.strptime(start_date, '%Y-%m-%d')
        end_dt = datetime.strptime(end_date, '%Y-%m-%d')

        if start_dt >= end_dt:
            return False, "End date must be after start date!"

        if start_dt < datetime.now():
            return False, "Start date cannot be in the past!"

        return True, ""
    except:
        return False, "Invalid date format!"
```

User Authentication

```
import hashlib
from datetime import datetime
from validation import validate_email, validate_egyptian_phone, validate_name
```

```
import data_storage as storage

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

def verify_password(stored_hash, password):
    return stored_hash == hash_password(password)

def create_user(first_name, last_name, email, password, mobile):
    user_data = {
        'first_name': first_name,
        'last_name': last_name,
        'email': email,
        'password_hash': hash_password(password),
        'mobile': mobile,
        'is_active': False,
        'created_at': datetime.now().isoformat()
    }
    return user_data

def register_user():
    print("\n=== REGISTRATION ===")

    first_name = input("First Name: ").strip()
    if not validate_name(first_name):
        print("❌ First name must contain only letters and spaces!")
        return False

    last_name = input("Last Name: ").strip()
    if not validate_name(last_name):
        print("❌ Last name must contain only letters and spaces!")
        return False

    email = input("Email: ").strip().lower()
    if not validate_email(email):
        print("❌ Invalid email format!")
```

```
        return False
    if email in storage.users_data:
        print("❌ Email already exists!")
        return False

    password = input("Password: ").strip()
    if len(password) < 6:
        print("❌ Password must be at least 6 characters!")
        return False

    confirm_password = input("Confirm Password: ").strip()
    if password != confirm_password:
        print("❌ Passwords don't match!")
        return False

    mobile = input("Mobile Phone: ").strip()
    if not validate_egyptian_phone(mobile):
        print("❌ Invalid Egyptian phone number format!")
        return False

    user = create_user(first_name, last_name, email, password, mobile)
    storage.users_data[email] = user
    storage.save_data()

    print("✅ Registration successful!")
    print("⚠️  Account created but not activated. Please activate your account t
o login.")

    activate = input("Activate account now? (y/n): ").lower()
    if activate == 'y':
        storage.users_data[email]['is_active'] = True
        storage.save_data()
        print("✅ Account activated!")

    return True
```

```
def login_user():
    print("\n=== LOGIN ===")

    email = input("Email: ").strip().lower()
    password = input("Password: ").strip()

    if email not in storage.users_data:
        print("❌ Email not found!")
        return False

    user = storage.users_data[email]

    if not user['is_active']:
        print("❌ Account not activated!")
        return False

    if not verify_password(user['password_hash'], password):
        print("❌ Invalid password!")
        return False

    storage.set_current_user(email)
    print(f"✅ Welcome, {user['first_name']} {user['last_name']}!")
    return True

def logout_user():
    storage.clear_current_user()
    print("✅ Logged out successfully!")
```

Project Manager

```
from datetime import datetime
from validation import validate_date, validate_date_range
import data_storage as storage

def create_project_data(title, details, total_target, start_date, end_date, owner_
```

```
email):
    return {
        'title': title,
        'details': details,
        'total_target': total_target,
        'start_date': start_date,
        'end_date': end_date,
        'owner_email': owner_email,
        'created_at': datetime.now().isoformat(),
        'current_amount': 0.0
    }

def create_new_project():
    current_user = storage.get_current_user()
    if not current_user:
        print("❌ Please login first!")
        return False

    print("\n=== CREATE PROJECT ===")

    title = input("Project Title: ").strip()
    if not title:
        print("❌ Title cannot be empty!")
        return False

    details = input("Project Details: ").strip()
    if not details:
        print("❌ Details cannot be empty!")
        return False

    try:
        total_target = float(input("Total Target (EGP): "))
        if total_target <= 0:
            print("❌ Target must be positive!")
            return False
    except ValueError:
```

```
        print("❌ Invalid target amount!")
        return False

    start_date = input("Start Date (YYYY-MM-DD): ").strip()
    if not validate_date(start_date):
        print("❌ Invalid start date format!")
        return False

    end_date = input("End Date (YYYY-MM-DD): ").strip()
    if not validate_date(end_date):
        print("❌ Invalid end date format!")
        return False

    is_valid, error_msg = validate_date_range(start_date, end_date)
    if not is_valid:
        print(f"❌ {error_msg}")
        return False

    project = create_project_data(title, details, total_target, start_date, end_date,
current_user['email'])
    storage.projects_data.append(project)
    storage.save_data()

    print("✅ Project created successfully!")
    return True

def view_all_projects():
    print("\n=== ALL PROJECTS ===")

    if not storage.projects_data:
        print("No projects available.")
        return

    for i, project in enumerate(storage.projects_data, 1):
        progress = (project['current_amount'] / project['total_target']) * 100
        print(f"\n{i}. {project['title']}")
```

```
        print(f"   Owner: {project['owner_email']}")
        print(f"   Details: {project['details']}")
        print(f"   Target: {project['total_target']:,.2f} EGP")
        print(f"   Progress: {project['current_amount']:,.2f} EGP ({progress:.1
f}%)")
        print(f"   Duration: {project['start_date']} to {project['end_date']}")

def get_user_projects(email):
    return [p for p in storage.projects_data if p['owner_email'] == email]

def view_my_projects():
    current_user = storage.get_current_user()
    if not current_user:
        print("❌ Please login first!")
        return

    print("\n=== MY PROJECTS ===")

    my_projects = get_user_projects(current_user['email'])

    if not my_projects:
        print("You haven't created any projects yet.")
        return

    for i, project in enumerate(my_projects, 1):
        progress = (project['current_amount'] / project['total_target']) * 100
        print(f"\n{i}. {project['title']}")
        print(f"   Details: {project['details']}")
        print(f"   Target: {project['total_target']:,.2f} EGP")
        print(f"   Progress: {project['current_amount']:,.2f} EGP ({progress:.1
f}%)")
        print(f"   Duration: {project['start_date']} to {project['end_date']}")

def edit_user_project():
    current_user = storage.get_current_user()
    if not current_user:
```

```
    print("❌ Please login first!")
    return False

my_projects = get_user_projects(current_user['email'])

if not my_projects:
    print("❌ You have no projects to edit!")
    return False

print("\n=== EDIT PROJECT ===")
print("Your projects:")
for i, project in enumerate(my_projects, 1):
    print(f"{i}. {project['title']}")

try:
    choice = int(input("Select project number to edit: ")) - 1
    if choice < 0 or choice >= len(my_projects):
        print("❌ Invalid selection!")
        return False
except ValueError:
    print("❌ Invalid input!")
    return False

project = my_projects[choice]

print(f"\nEditing: {project['title']}")
print("Press Enter to keep current value.")

new_title = input(f"Title ({project['title']}): ").strip()
if new_title:
    project['title'] = new_title

new_details = input(f"Details ({project['details']}): ").strip()
if new_details:
    project['details'] = new_details
```

```
    new_target = input(f"Target ({project['total_target']}): ").strip()
    if new_target:
        try:
            target_value = float(new_target)
            if target_value > 0:
                project['total_target'] = target_value
            else:
                print("⚠️ Invalid target, keeping current value.")
        except ValueError:
            print("⚠️ Invalid target format, keeping current value.")

    storage.save_data()
    print("✅ Project updated successfully!")
    return True

def delete_user_project():
    current_user = storage.get_current_user()
    if not current_user:
        print("❌ Please login first!")
        return False

    my_projects = get_user_projects(current_user['email'])

    if not my_projects:
        print("❌ You have no projects to delete!")
        return False

    print("\n=== DELETE PROJECT ===")
    print("Your projects:")
    for i, project in enumerate(my_projects, 1):
        print(f"{i}. {project['title']}")

    try:
        choice = int(input("Select project number to delete: ")) - 1
        if choice < 0 or choice >= len(my_projects):
            print("❌ Invalid selection!")
```

```
            return False
    except ValueError:
        print("❌ Invalid input!")
        return False

    project = my_projects[choice]

    confirm = input(f"Are you sure you want to delete '{project['title']}'? (y/N): ").lower()
    if confirm == 'y':
        storage.projects_data.remove(project)
        storage.save_data()
        print("✅ Project deleted successfully!")
        return True
    else:
        print("❌ Deletion cancelled.")
        return False

def search_projects_by_date():
    print("\n=== SEARCH PROJECTS BY DATE ===")

    date_str = input("Enter date (YYYY-MM-DD): ").strip()
    if not validate_date(date_str):
        print("❌ Invalid date format!")
        return

    search_date = datetime.strptime(date_str, '%Y-%m-%d').date()

    matching_projects = []
    for project in storage.projects_data:
        start_date = datetime.strptime(project['start_date'], '%Y-%m-%d').date()
        end_date = datetime.strptime(project['end_date'], '%Y-%m-%d').date()

        if start_date <= search_date <= end_date:
            matching_projects.append(project)
```

```
    if not matching_projects:
        print(f"No projects found running on {date_str}")
        return

    print(f"\nProjects running on {date_str}:")
    for i, project in enumerate(matching_projects, 1):
        progress = (project['current_amount'] / project['total_target']) * 100
        print(f"\n{i}. {project['title']}")
        print(f"   Owner: {project['owner_email']}")
        print(f"   Details: {project['details']}")
        print(f"   Target: {project['total_target']:,.2f} EGP")
        print(f"   Progress: {project['current_amount']:,.2f} EGP ({progress:.1
f}%)")
        print(f"   Duration: {project['start_date']} to {project['end_date']}")
```

Main

```
import data_storage as storage
from user_auth import register_user, login_user, logout_user
from project_manager import (
    create_new_project, view_all_projects, view_my_projects,
    edit_user_project, delete_user_project, search_projects_by_date
)

def show_guest_menu():
    print("\n=== MAIN MENU ===")
    print("1. Register")
    print("2. Login")
    print("3. View All Projects")
    print("4. Search Projects by Date")
    print("0. Exit")

def show_user_menu():
    current_user = storage.get_current_user()
    print(f"\n=== USER MENU ({current_user['first_name']}) ===")
```

```
    print("1. Create Project")
    print("2. View All Projects")
    print("3. View My Projects")
    print("4. Edit My Project")
    print("5. Delete My Project")
    print("6. Search Projects by Date")
    print("7. Logout")
    print("0. Exit")

def handle_guest_menu(choice):
    if choice == '1':
        register_user()
    elif choice == '2':
        login_user()
    elif choice == '3':
        view_all_projects()
    elif choice == '4':
        search_projects_by_date()
    elif choice == '0':
        return False
    else:
        print("❌ Invalid option!")
    return True

def handle_user_menu(choice):
    if choice == '1':
        create_new_project()
    elif choice == '2':
        view_all_projects()
    elif choice == '3':
        view_my_projects()
    elif choice == '4':
        edit_user_project()
    elif choice == '5':
        delete_user_project()
    elif choice == '6':
```

```
        search_projects_by_date()
    elif choice == '7':
        logout_user()
    elif choice == '0':
        return False
    else:
        print("❌ Invalid option!")
    return True

def main():
    # Initialize data
    storage.load_data()

    print("🎉 Welcome to the Crowdfunding Platform!")

    while True:
        current_user = storage.get_current_user()

        if not current_user:
            show_guest_menu()
            choice = input("\nSelect option: ").strip()
            if not handle_guest_menu(choice):
                break
        else:
            show_user_menu()
            choice = input("\nSelect option: ").strip()
            if not handle_user_menu(choice):
                break

    print("👋 Goodbye!")

if __name__ == "__main__":
    main()
```

Run Compiler

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS                    Python Debug Console

\AI & Data_Science\ITI Tasks\Python\crowdfunding_app\crowdfunding_app.py'
🎉 Welcome to the Crowdfunding Platform!

=== MAIN MENU ===
1. Register
2. Login
3. View All Projects
4. Search Projects by Date
0. Exit

Select option: 1

=== REGISTRATION ===
First Name: Alaa
Last Name: Reab21
❌ Last name must contain only letters and spaces!

=== MAIN MENU ===
1. Register
2. Login
3. View All Projects
4. Search Projects by Date
0. Exit

Select option: 1

=== REGISTRATION ===
First Name: Alaa
Last Name: Rehab
Email: alaar@gmail.com
Password: Alaa231
Confirm Password: Alaa231
Mobile Phone: 01223974535
✅ Registration successful!
⚠️ Account created but not activated. Please activate your account to login.
Activate account now? (y/n): y
✅ Account activated!
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS                    Python Debug Console

=== MAIN MENU ===
1. Register
2. Login
3. View All Projects
4. Search Projects by Date
0. Exit

Select option: 2

=== LOGIN ===
Email: alaar@gmail.com
Password: Alaa231
✅ Welcome, Alaa Rehab!

=== USER MENU (Alaa) ===
1. Create Project
2. View All Projects
3. View My Projects
4. Edit My Project
5. Delete My Project
6. Search Projects by Date
7. Logout
0. Exit

Select option: 1

=== CREATE PROJECT ===
Project Title: Health Care
Project Details: AI-powered predictive system for chronic diseases (Hypertension, Stroke, Diabetes) using real heal
thcare data. Includes ML models, Streamlit interface.
Total Target (EGP): 100000
Start Date (YYYY-MM-DD): 2025-8-20
End Date (YYYY-MM-DD): 2025-9-30
✅ Project created successfully!

=== USER MENU (Alaa) ===
1. Create Project
```

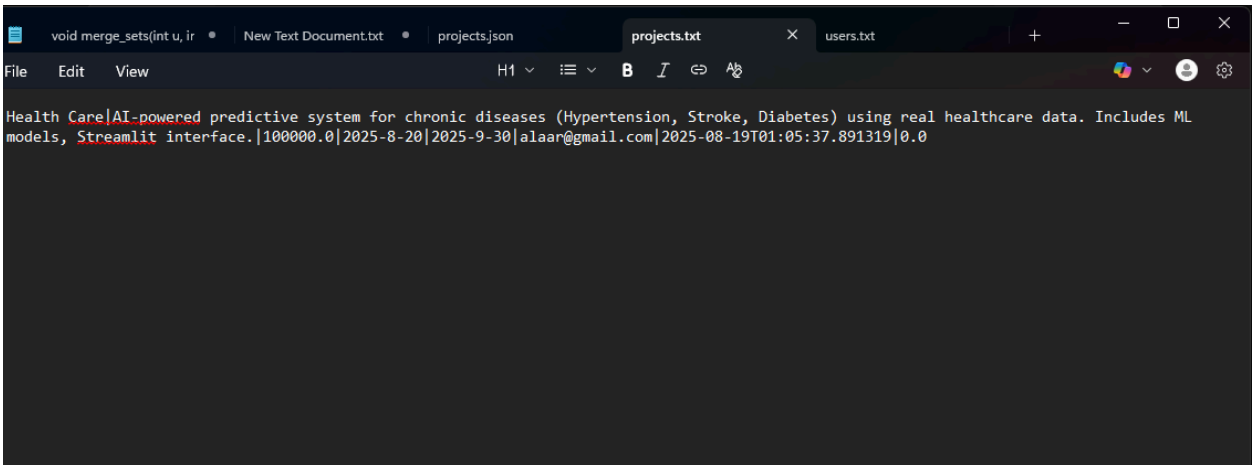
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS        Python Debug Console

=== USER MENU (Alaa) ===
1. Create Project
2. View All Projects
3. View My Projects
4. Edit My Project
5. Delete My Project
6. Search Projects by Date
7. Logout
0. Exit

Select option: 2

=== ALL PROJECTS ===

1. Health Care
   Owner: alaar@gmail.com
   Details: AI-powered predictive system for chronic diseases (Hypertension, Stroke, Diabetes) using real healthcar
e data. Includes ML models, Streamlit interface.
   Target: 100,000.00 EGP
   Progress: 0.00 EGP (0.0%)
   Duration: 2025-8-20 to 2025-9-30

=== USER MENU (Alaa) ===
1. Create Project
2. View All Projects
3. View My Projects
4. Edit My Project
5. Delete My Project
6. Search Projects by Date
7. Logout
0. Exit

Select option: 7
✅ Logged out successfully!

=== MAIN MENU ===
1. Register
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS        Python Debug Console

3. View My Projects
4. Edit My Project
5. Delete My Project
6. Search Projects by Date
7. Logout
0. Exit

Select option: 7
✅ Logged out successfully!

=== MAIN MENU ===
1. Register
2. Login
3. View All Projects
4. Search Projects by Date
0. Exit

Select option: 0
👋 Goodbye!
PS E:\AI & Data_Science\ITI Tasks>
```

Users File

```
void merge_sets(int u, ir   New Text Document.txt   projects.json   projects.txt   users.txt
File   Edit   View   H1

alaar@gmail.com|Alaa|Rehab|611ef89149b549de430f34ca0038417bbd9001d36ddb0c70ce5cfb3849ae7824|01223974535|True|2025-08-19T01:04:10.310467
```

Projects File


void merge_sets(int u, ir
New Text Document.txt
projects.json
projects.txt
users.txt
File
Edit
View
H1
Health Care|AI-powered predictive system for chronic diseases (Hypertension, Stroke, Diabetes) using real healthcare data. Includes ML models, Streamlit interface.|100000.0|2025-8-20|2025-9-30|alaar@gmail.com|2025-08-19T01:05:37.891319|0.0